Herzberg, Dominikus

Embodied Cognition als Lehrparadigma in der Programmierausbildung

Schmohl, Tobias [Hrsg.]; To, Kieu-Anh [Hrsg.]: Hochschullehre als reflektierte Praxis. Fachdidaktische Fallbeispiele mit Transferpotenzial. 2., vollständig überarbeitete und erweiterte Auflage. Bielefeld : wbv 2019, S. 29-41. - (TeachingXchange; 1)

Hochschullehre als reflektierte Praxis

Fachdidaktische Fallbeispiele mit Transferpotenzial

Tobias Schmohl, Kieu-Anh To (Hg.)

Embodied Cognition als Lehrparadigma in der Programmierausbildung

Dominikus Herzberg

Abstract

Dieser Beitrag stellt den aus der Kognitionswissenschaft stammenden Ansatz des Embodied Cognition vor und verargumentiert sein Potenzial für die Lehre und die Lehr-Lernforschung. Am Beispiel der hochschulischen Programmierausbildung wird aufgezeigt, wie ein kognitionswissenschaftlicher Ansatz zu einem didaktischen Instrument führt, der Graph-Metapher, welche die in der Programmierung übliche Objekt-Orientierung stringent vermittelbar und systematisch und schlüssig erklärbar macht – dies vor allem auch in der Hinsicht, dass informatische Modellbildung und lebensweltliche Erfahrungen reflektiert und in ihren Unterschieden erkannt werden. Darüber hinaus wird eine theoretische Verortung im Neuen Realismus vorgestellt, die der Lerntheorie des Kognitivismus als Paradigma eine etwas andere Färbung verleiht.

Schlagworte: Kognitionswissenschaft, embodied cognition, objektorientierte Programmierung, Programmierausbildung, Didaktik

1 Einleitung

Diese Arbeit möchte für den Kognitivismus, der zu den drei großen Lerntheorien zählt, eine Variante zur Diskussion stellen und ein Potenzial für die Lehre und die Lehr-Lernforschung verargumentieren, das speziell eingeht auf die Problematik der Programmierausbildung von Erstsemestern in Informatik-Studiengängen. Die Theorie der Embodied Cognition, von der die Rede sein wird, kann ontologisch und epistemologisch mit Hilfe des Neuen Realismus unterfüttert werden – damit deutet sich an, dass Embodied Cognition der Lerntheorie des Kognitivismus als Paradigma eine leicht andere Färbung verleiht.

Ausgangspunkt dieser Arbeit ist eine Vorarbeit zur Mathematik, deren ideengeschichtliche Entwicklung als eine Folge metaphorischer Mappings verstanden und in den Dienst der Didaktik gestellt werden kann: das Buch *„Where Mathematics comes from“* von Lakoff und Núñez (2000), das die Metaphern und Mappings für die Arithmetik, die Algebra, die Logik, die Mengenlehre und die Zahlenlehre bis hin zum Begriff der Unendlichkeit durchdekliniert. Viele der Metaphern und Mappings sind über Hunderte von Jahren entstanden und haben ihre Bewährung lange hinter

sich. Das Bemerkenswerte ist, dass sie aus der Perspektive der Embodied Cognition allesamt verkörperte Ursprünge haben, die als solche selten erkannt und verstanden werden. Dabei besteht genau darin ihr didaktischer Wert. Man kann aus den Metaphern und ihren Mappings didaktische Hilfsmittel und Vorgehensweisen ableiten und sie der Lehr-Lernforschung zuführen.

Aufgrund der Nähe der Informatik zur Mathematik nutze ich das Buch von Lakoff und Núñez (2000) als Setzung für eine Anwendung auf die Herausforderungen der Vermittlung der objektorientierten Programmierung; ich arbeite also direkt mit den Erkenntnissen der beiden Autoren und stelle anhand konkreter Beispiele vor, wie sich Metaphern in der informatischen Modellbildung darstellen.

2 Die Schwierigkeit, das Programmieren zu erlernen

Die Studienabbruchquoten in der Informatik gehören zu den höchsten überhaupt. An Universitäten brachen 45 Prozent (2014) bzw. 46 Prozent (2016) der Absolventen das Bachelor-Studium der Informatik ab (nur übertroffen von der Mathematik); an Fachhochschulen waren es 41 Prozent (2014) bzw. 39 Prozent (2016), hier nur übertroffen von der Elektrotechnik (Heublein und Schmelzer 2018, S. 25 f.). Kritisch sind die ersten beiden Semester – vor allem dann, wenn Leistungsprobleme eine große Rolle spielen (Heublein et al. 2017, S. 34). Abseits von weiteren möglichen Ursachen für den Studienabbruch kann man die Frage stellen, ob es etwas gibt, das die Informatik inhärent schwierig zu vermitteln und zu erlernen macht – der Verdacht scheint nicht ganz unbegründet.

Einen entscheidenden Anteil am Informatikstudium hat die Programmierausbildung in den ersten beiden Semestern. In der Tat ist die Programmierausbildung ein problembehaftetes Fach, wie es die vielen Publikationen zum Thema belegen. Ein Literatur-Review von Qian und Lehman (2017) befasst sich mit den falschen Vorstellungen, die Studienanfänger gegenüber der Programmierung haben, und macht drei Kategorien des Programmierverständnisses aus: Wissen um die Syntaktik, also grundlegende Kenntnisse um den Aufbau und die Regeln einer Programmiersprache; Konzeptwissen, womit ein Verständnis um Programmierkonstrukte und -prinzipien gemeint ist, und schließlich Entwicklungswissen, was die Anwendung von Syntax- und Konzeptwissen zur Lösung von neuen Programmierproblemen meint (S. 3 f.). Für die Verständnisschwierigkeiten beim Erlernen einer Programmiersprache werden mehrere Faktoren identifiziert: „unfamiliarity of syntax, natural language, math knowledge, inaccurate mental models, lack of strategies, programming environments, inappropriate instruction, and teachers' knowledge" (S. 17). Bemerkenswerterweise können auch Lehrende zu den Schwierigkeiten beitragen, indem sie unpassende Analogien, Modellvorstellungen und Metaphern nutzen (S. 10) – und genau an dieser Stelle hat die vorliegende Arbeit ihren Ansatzpunkt, was ebenso Anschluss nimmt an die „inaccurate mental models" aufseiten der Studierenden.

Im Lichte welcher der drei großen Lerntheorien werden solche Probleme in der Programmierausbildung angegangen und reflektiert? Obgleich ein entsprechendes Bewusstsein für alle drei Lerntheorien besteht (vgl. Quevedo-Torrero, 2009), scheint die Informatik eine Präferenz für den Konstruktivismus als lerntheoretische Fundierung zu haben (siehe z. B. Troelstra, 1999 und Ben-Ari, 2001). Im Rahmen seiner Überlegungen grenzt Ben-Ari (2001) die Anwendung des Konstruktivismus in der Informatikausbildung ab von den Naturwissenschaften, indem er zwei Besonderheiten für die Informatik feststellt (S. 56, Hervorhebung im Original):

1. „A (beginning) computer scientist student has no *effective model* of a computer."
2. „The computer forms an accessible *ontological reality*."

Beides sind gültige und zutreffende Punkte. Ein zum Konstruktivismus alternativer Vorschlag muss auf diese beiden Punkte eingehen. Die Kombination aus Neuem Realismus und Embodied Cognition scheint mir das zu leisten.

3 Der Neue Realismus und Embodied Cognition

Der Neue Realismus steht in Opposition zum Konstruktivismus und geht davon aus, „dass wir die Welt so erkennen, wie sie an sich ist", was nicht ausschließt, sich auch täuschen zu können (Gabriel 2018, S. 11). Ich lege hier die Arbeiten des Philosophen Markus Gabriel zugrunde.

Eine kurze Skizze zum Neuen Realismus, welcher der analytischen Erkenntnistheorie eine synthetische gegenüberstellt (Gabriel, 2013, S. 37 f.): Die ontologische Grundeinheit des Neuen Realismus sind sogenannte Sinnfelder – ein Sinnfeld ist der „Ort", an dem etwas erscheint (Gabriel, 2018, S. 68). „Zu existieren besteht demnach darin, in einem Sinnfeld zu erscheinen" (Gabriel & Krüger, 2018, S. 76). In einem Sinnfeld können Gegenstände und/oder Gegenstandsbereiche erscheinen; sie sind damit existent, wobei Sinn die Art bezeichnet, wie ein Gegenstand erscheint (Gabriel, 2018, S. 91). Es ist der Sinn, der Sinnfelder voneinander unterscheidet (S. 113).

Ein Gegenstand ist etwas, „worauf man sich mit wahrheitsfähigen Gedanken beziehen kann" (Gabriel, 2013, S. 231). Ein Gegenstandsbereich ist ein Bereich, „der eine bestimmte Art von Gegenständen enthält, wobei Regeln feststehen, die diese Gegenstände miteinander verbinden" (Gabriel, 2018, S. 35).

Zurück zu den Sinnfeldern: Welche Sinnfelder es gibt, „ist keine Leistung der Ontologie, sondern der empirischen Wissenschaften" (S. 114) oder schlicht der Erfahrung. Allerdings geht es stets um Gegenstandsbereiche. „Die Wissenschaft erkennt auf eine Weise, die für jeden nachvollziehbar und überprüfbar ist, der sich ihre Methoden angeeignet hat" (S. 132). Erweist sich ein Gegenstandsbereich als Redebereich (auf gut Deutsch: als „Geschwätz"), nimmt man eine ontologische Reduktion vor; deshalb braucht man für viele Gegenstandsbereiche eine Irrtumstheorie (S. 54).

Ich halte diese Interpretation des Neuen Realismus als wissenschaftstheoretische Grundlage tragfähig für die Vorstellungen und Annahmen der „verkörperten Kognition", die Begriffe wie „Metapher" und „Frame" einführt und damit kognitive Konzepte in Bezug zu Gegenstandsbereichen bringt, die in Sinnfeldern erscheinen können.

In der Kognitionswissenschaft gibt es die Theorie der *Embodied Cognition*, die davon ausgeht, dass das Denken nicht nur einen Köper voraussetzt, sondern vielmehr durch die Wahrnehmung und Motorik und die Interaktion mit und in der Welt ebenso bedingt wie auch begrenzt ist. Alles Denkbare muss eine Abbildung in verkörperten Ausdrucks- und Sinnesformen finden; daher der Begriff der „verkörperten Kognition", im Deutschen oft nur als „Embodiment" bezeichnet.

Interessant ist die Verbindung mit der Kognitiven Linguistik, wie sie auf den Linguisten George Lakoff und den Philosophen Mark L. Johnson zurückgeht (Lakoff & Johnson, 1999). Im Zentrum steht die Frage, wie Worte ihre Bedeutung und ihren Sinn erhalten und welche Rolle dabei Metaphern und gedankliche Simulationen spielen (Bergen, 2012). Die Sprach- und Kognitionswissenschaftlerin Elisabeth Wehling zielt in ihrem Buch „Politisches Framing" (Wehling, 2016) zwar auf das politische Denken ab (ein Anliegen, das sie mit ihrem Lehrer Lakoff teilt), liefert darin jedoch eine prägnante Zusammenfassung zum Forschungsstand. Das Wichtigste in Kürze: Menschen begreifen Worte, indem das Gehirn die mit den Worten assoziierten Bewegungsabläufe, Sinneseindrücke und Gefühle simuliert (S. 21 f.). Außerdem aktiviert es einen Deutungsrahmen, *Frame* genannt, der körperliche, sprachliche und kulturelle Erfahrungen beinhaltet (S. 28); auch diese Anteile werden bei der Simulation mitaktiviert (S. 30). Frames beeinflussen nicht nur die Sprachverarbeitung, sondern auch die Wahrnehmung (S. 32). Entscheidend für den Kontext der Lehr-Lernforschung ist, dass abstrakte Konzepte und Ideen stets „über Metaphern an körperliche Erfahrungen angebunden und damit 'denkbar' gemacht" werden, was man als *Metaphoric Mapping* bezeichnet (S. 68). Genau genommen geht es um konzeptuelle Metaphern, die unweigerlich unser Alltagsdenken strukturieren und sich in sprachlichen Metaphern ausdrücken (S. 69 f.). Sprache ist immer durch Metaphern strukturiert, weil Menschen automatisch in konzeptuellen Metaphern denken (S. 71). Metaphorische Mappings erlauben es uns, konkrete Welterfahrung mit abstrakten Ideen und Konzepten zu koppeln (S. 71). Damit ist die *Conceptual Metaphor Theory* skizziert.

Experimentell ist die Theorie mittlerweile gut untermauert, vgl. Bergen (2012). Die Existenz von Frames und die durch Worte ausgelösten Simulationen im Gehirn passen zur ontologischen und epistemologischen Sicht des Neuen Realismus. Das Embodiment tariert den Bezug von Gehirn, Körper und Welt neu aus: „Die Aufgabe, Probleme zu lösen, muss nicht allein vom Gehirn erfüllt werden, sondern kann zwischen Gehirn, Körper und Welt aufgeteilt werden" (Crawford, 2016). Ich möchte aufzeigen, dass man mit der Conceptual Metaphor Theory zu interessanten Lehrkonzepten kommt, die notwendig im Sinne der Theorie des Embodiment auch dem Lernenden zugute kommen sollten.

4 Metaphern in der informatischen Modellbildung

In diesem Kapitel möchte ich an Beispielen aufzeigen, wie sich der Ansatz der Embodied Cognition mit Bezug auf die Arbeit von Lakoff und Núñez (2000) in der Informatik für die Programmierausbildung anwenden lässt. Dazu ist Lehrmaterial in Form eines Textes zur Objektorientierung entstanden – einem Programmierparadigma, das Studierenden notorisch Probleme bereitet. Der Text ist in Teilen eingeflochten, durch einen serifenlosen Font hervorgehoben und zeichnet sich durch einen informellen Stil aus.

4.1 Objektorientierung als Konzeption

Das Problem des Erlernens der Objektorientierung beginnt bereits mit dem Namen. Mit welchen Vorstellungen ist das Wort „Objekt“ assoziiert, welcher Frame wird aktiviert? Die Wortbedeutung gibt einen ersten Hinweis; laut Duden bedeutet „Objekt“ allgemein „Gegenstand, auf den das Interesse, das Denken, das Handeln gerichtet ist.“ Eine erste Annäherung zum aktivierten Frame liefern Synonyme und typische Wortverbindungen. Wieder hilft in beiden Fällen der Duden weiter. Zu dem Wort „Objekt“ gibt es die folgenden Synonyme:

- Artikel, Ding, Erzeugnis, Etwas, Gegenstand, Körper, Produkt, Sache; (umgangssprachlich) Teil
- Stoff, Thema, Thematik, Themenstellung; (bildungssprachlich) Sujet
- Grundstück, Haus; (Wirtschaft) Immobilie

Mit dem Wort „Objekt“ sind die Adjektive *ausgestellt*, *bewegend*, *bewegt*, *dreidimensional*, *einzeln* und weniger stark *obskur* verbunden; auch die Substantive *Zeichnung*, *Malerei*, *Klasse* und weniger stark *Bild* und *Installation* und *Subjekt* sind mit ihm verbunden.

Noch besser wäre es, Studierende nach ihrem Verständnis des Begriffs „Objekt“ zu befragen, aber es deutet sich an dieser Stelle an, dass dieser Begriff als abstrakte Metapher für einen Gegenstand herhalten kann und vermutlich so in erster Annäherung verstanden wird. Dass Gegenstände einen Zustand haben, mag noch anschlussfähig an die Metapher sein; auch mag es noch verträglich mit der Metapher sein, dass Gegenstände gleichen Typs einer gemeinsamen „Klasse“ angehören – aber mit der Zunahme an objektorientierten Konzeptbegriffen und Aussagen kollidieren die Metaphern und ihre Frames. „Jedes Objekt ist die Instanz einer Klasse“ – diese korrekte Aussage ist schwer verständlich, zumal der Begriff der „Instanz“, wenn er überhaupt verstanden wird, Synonyme wie „Administration, Amt, Behörde, Dienststelle, Institution, Stelle, Verwaltung, [Verwaltungs]organ“ als Frame-Annäherung mit sich trägt. Die objektorientierte Konzeption der „Vererbung“ erinnert an die „Weitergabe von Erbanlagen von einer Generation an die folgende“ (Duden) – den tatsächlichen Sachverhalt kann ein Container-Bildschema in den mengentheoretischen Bezügen hier deutlich besser vermitteln als der biologische Bezug.

Diese allein durch die Begriffe erzeugte Metaphern-Verwirrung ist sicherlich ein Anzeichen einer noch jungen Disziplin. Die Ideenwelt der Informatik hat nicht

einmal zwei Generationen hinter sich bringen können, um die Tragfähigkeit und didaktische Tauglichkeit von Metaphern auszuloten und gegebenenfalls neue Metaphern ins Spiel zu bringen. Die andere Seite der Medaille ist, dass eine Computersprache und ein Programmierparadigma eine ontologische Setzung vornehmen, deren begrifflich-metaphorische Einkleidung vielleicht misslungen sein mag, die sich aber operativ als axiomatische Setzung versteht und im Sinnfeld einer Programmiersprache erscheint.

Dazu gibt es eine Lösung, die Lakoff und Núñez (2000) unter der Idee „Mengen sind Graph-Metaphern" einbringen (S. 146–152). Statisch typisierte, objektorientierte Programmiersprachen basieren formal auf einem mengentheoretischen Konzept, das unmittelbar von der Graph-Metapher profitiert. Die Graph-Metapher macht ...

1. Objekte als graphische Einheiten einfach fassbar,
2. das für Studierende schwer verständliche (da ansonsten und auch in der Interaktion mit der Sprache unsichtbare) Konzept der Referenz als Pfeil anschaulich und leicht nachvollziehbar,
3. das Konzept der Klasse zur Aussage darüber, welche Pfeile man von einem Objekt dieser Klasse zu welchen anderen Objekten zeichnen darf.

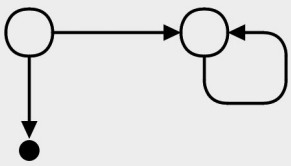
Zur Erklärung folgt ein Auszug aus dem von mir entwickelten Lehrmaterial, das sich in einem informellen Stil präsentiert und zur besseren Abgrenzung in Kästen mit grauem Hintergrund dargestellt ist.

Kästchen sind Objekte, Pfeile sind Verweise

Was auch immer ein Objekt sein soll. Stellen Sie sich vor, dass Sie auf einem Papier mit einem Stift Kreise oder Kästchen zeichnen. Diese Kreise oder Kästchen nennen wir Objekte; wichtig ist, dass sich die Kreise nicht überschneiden und auch nicht ineinander verschachtelt sind.

Die Kreise können wir mit Pfeilen verbinden. Ein Pfeil beginnt am Kreisrand des einen Kreises und endet mit einer Pfeilspitze am Kreisrand eines anderen Kreises; grundsätzlich denkbar ist aber auch, dass ein Pfeil vom gleichen Kreis ausgeht und auch wieder bei ihm endet. Die Pfeile nennen wir Verweise oder auch Referenzen.

Wenn wir einen Pfeil explizit *nicht* bei einem anderen Kreis enden lassen wollen, dann lassen wir den Pfeil ein wenig ins Leere laufen und bringen an der Pfeilspitze einen kleinen, dicken Knubbel an. Diesen ins Leere laufenden Pfeil nennen wir einen „Null-Verweis" oder auch „Null-Referenz".



Es macht übrigens nichts, wenn sich die Linien der Pfeile überschneiden. Allerdings müssen wir Namen an die Pfeile schreiben, sonst wissen wir nicht, welcher Pfeil wofür ist. Dazu gleich mehr.

Klassen definieren Objektarten

Was nun noch hinzukommt, ist, dass wir verschiedene Arten von Objekten unterscheiden – der Fachmensch spricht von Klassen. Diesen verschiedenen Objektarten geben wir Namen. Damit wir wissen, von welcher Art ein Objekt ist, schreiben wir den Namen in das Objekt hinein.

Eine Objektart, eine Klasse, hat nicht nur einen Namen. Mit der Objektart ist festgelegt, wie viele Pfeile von einem Objekt dieser Art ausgehen (es ist eine feste Anzahl), wie diese Pfeile heißen (darum müssen wir die Pfeile beschriften) und bei was für einer Art von Objekt der Pfeil enden muss, wenn er kein Null-Verweis sein möchte.

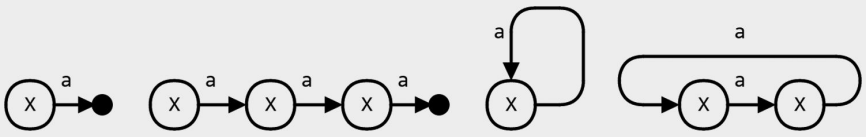
Ein Beispiel: Wir legen für Objekte der Art, also der Klasse „X“, fest, dass jedes Objekt dieser Klasse einen Pfeil haben muss, der den Namen „a“ hat und auf ein Objekt der Klasse „X“ zeigt. Man könnte das sehr schematisch wie folgt aufschreiben:

> Objektart „X“ hat einen Pfeil namens „a“, der bei einem Objekt der Art „X“ enden muss, sofern der Pfeil kein Null-Verweis ist.

Oder noch kürzer und formaler:

```
class X {
      X a;
}
```

Wie könnten mögliche Objektbilder dazu aussehen? Versuchen Sie, ein paar Objekt-Pfeil-Diagramme zu zeichnen!



Die Graph-Metapher hilft sehr gut dabei, den Begriff der Datenstruktur anzubringen. Graphen bilden Strukturen ab, und dass es sich bei den Objekten nicht um Gegenstände, sondern um gegenstandsfreie Daten handelt („Kreise“ in den Bildern), ist leicht einsichtig. Die Graph-Metapher richtet den Blick außerdem auf den entscheidenden und wichtigen Punkt: Die Klasse ist der Ort, an dem Namen für die Klassen vergeben werden. Der Klassenname wird zum Träger einer Metapher für die damit aufsetzbaren Datenstrukturen. Darauf geht der folgende Abschnitt am Beispiel des Stapels ein.

4.2 Objektorientierung zur Implementierung von Datenstrukturen

Viele Datenstrukturen tragen sehr sprechende Namen wie „Liste“, „Baum“ oder „Stapel“. Die Namen dienen als Metaphern, weil Struktur und Gebrauch an ihr Vorbild aus der Realwelt erinnern. Doch ist die Umsetzung einer Datenstruktur mit Schwierigkeiten behaftet, da es sich um einen Modellierungsvorgang handelt, der sämtliche physikalischen Grundlagen ignoriert und physikalische Bezüge, die die Metapher nahelegt, in abstrakte übersetzt.

Sie wissen, was ein Stapel (*stack*) ist. Nehmen wir einen Bücherstapel. Ein Buch liegt auf dem anderen, die Bücher schichten sich aufeinander. Das unmittelbar zugreifbare Buch liegt obenauf. Möchte man auf ein Buch darunter zugreifen, muss man mehrere Bücher von oben entfernen.

Der Stapel ist eine beliebte Metapher für eine Objektstruktur, die das Organisationsprinzip übernehmen soll. Damit stellt sich aber die Frage: Was ist ein Stapel?

Modellbildung, 1. Teil: Das Stapel-Prinzip

Den Stapel an sich gibt es nicht. Es ist ein Konzept, das eine Organisationsform beschreibt: Dinge liegen aufeinander. Von einem Stapel erwartet man, dass die aufeinander gelegten Dinge sich selber tragen und mit zunehmender Höhe einem Turm zu gleichen beginnen. Der Stapel will nichts konstruieren, er will keine Wand und er will kein Turm sein. Es geht mehr um die praktische Ausnutzung von wenig verfügbarem Platz.

Man sieht, man kommt mit der Metapher schnell an ihre Grenzen. Es ist schwer zu klären, was ein Stapel genau ist. Erinnern wir uns also an unser Vorhaben: Wir wollen den Stapel als Organisationsprinzip aufgreifen und nutzbar machen.

Das Übereinanderlegen funktioniert dank Schwerkraft und dank geeigneter Form der aufeinander liegenden Dinge. In unseren Objektwelten spielt beides keine Rolle. Wir können einen Pfeil für das Verhältnis *on top of* nutzen.

Starten wir einen ersten Versuch und wenden das Organisationsprinzip auf Objekte von der Art „Book“ (Buch) an.

```
class Book {
      Book onTopOf;
}
class View {
      Book stack;
}
```

Mit der „View“[1] kommt schön zum Ausdruck, dass wir einen Stapel sehen, es aber kein Konstrukt der Idee eines Stapels gibt, nur eine Nachbildung der Eigenschaft eines Buchs, auf einem anderen Buch liegen zu können. Der mit „stack“ gezeigte Blick sieht das oberste Buch und kann entlang der Pfeile die aufeinander verweisenden Bücher „sehen“.

Wir haben diese Struktur so ähnlich schon mit der Klasse „X“ gehabt. Streng genommen fehlt uns hier eine Beschränkung: Es sind keine Zyklen von „onTopOf“ erlaubt, die die Idee des Aufeinanderliegens *ad absurdum* führen.

Was ist der Nachteil dieser Struktur? Jede neue Objektart, die man stapeln möchte, muss den „onTopOf“-Pfeil haben. Wäre es nicht angebracht, die gestapelte Organisationsform zu isolieren von der Art der Objekte, die gestapelt werden?

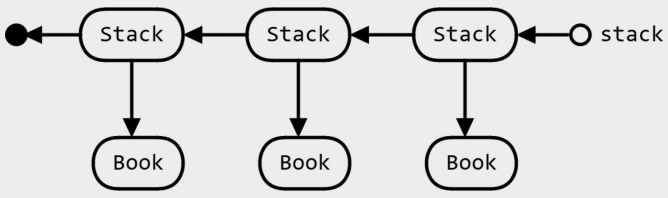
Ein neuer Versuch:

```
class Book { }

class Stack {
      Stack onTopOf;
      Book item;
}

class View {
      Stack stack;
}
```

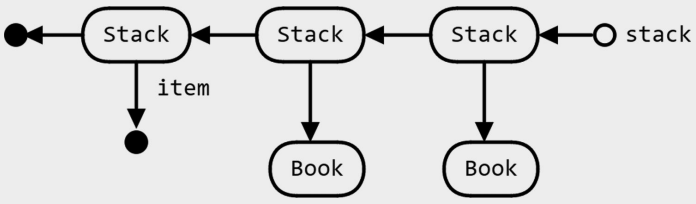
In dem Beispiel kann man sehen, dass die „Stack“-Objekte so etwas wie die Trägerstruktur für den Stapel bilden, hier wird das „Aufeinanderliegen“ abgebildet. Wie das Bild zeigt, hat die Darstellung in ihrer Ausrichtung von rechts nach links längst das Thema Schwerkraft und die Raumorganisation von „oben“ und „unten“ eingebüßt und modelliert das als eine Kette von „onTopOf“-Pfeilen. Die Bücher sind als Dinge (*items*) an die organisierende Trägerstruktur „angehängt“.



1 Die Einführung zu Views habe ich hier nicht aufgenommen.

Modellbildung, 2. Teil: Was ist ein leerer Stapel?
Es gibt sie nicht wirklich, die Idee des leeren Stapels. Wenn jemand mit dem Finger auf eine freie Stelle auf einem Tisch zeigt und sagt: „Da ist ein leerer Bücherstapel“, dann ist das sehr irritierend. Es ist eine merkwürde Aussage, etwas als nicht vorhanden zu bezeichnen, was genau aus diesem Grund nicht da ist. Wenn man von einem leeren Stapel spricht, dann ist das eher mit einem leeren Parkplatz vergleichbar. Der Parkplatz ist eine Fläche, die zur Besetzung durch ein Auto reserviert ist. So in etwa kann man sich die Idee des leeren Stapels vorstellen: Eine Fläche, die für einen Stapel reserviert ist. Und dann ist auch klar, dass mit nur einem Buch auf dieser Fläche der Anfang eines Bücherstapels gemacht ist.

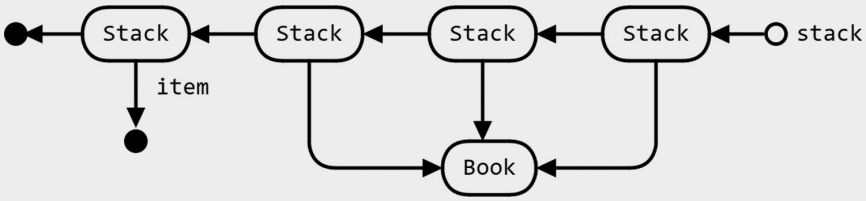
In diesem Sinne stellt der leere Stapel lediglich einen Anfang dar: Seht her, ich bin der Ort, an dem man Bücher aufeinanderstapeln kann. Diese Signalwirkung kann man darüber erreichen, dass wir eine Vereinbarung treffen: Die Existenz eines Stapels ist damit angelegt, indem wir ein „Stack“-Objekt anlegen, das weder „onTopOf“ irgendwas ist noch ein „item“ hat. Sprich: Die beiden Pfeile sind „null“-Verweise. So stellt sich das nachstehende Bild dar als ein Stapel mit zwei Büchern, die auf einem Ausgangspunkt zum Stapeln (das „Stack“-Objekt ganz links) aufgesetzt sind. Der „stack“-Verweis schaut auf das oberste Element des Stapels.



Diese Beschreibung eines Stapels in der Kunstwelt der Objekte ist eine extreme Kondensation eines Organisationsprinzips. Wir benötigen dazu eine einzige Objektart namens „Stack“, die das Rückgrat eines Stapels bildet. Mit einem Stapel, so wie wir ihn aus der „Echtwelt“ kennen, hat das nur rudimentär zu tun.

Es sollte deutlich geworden sein, wie unpassend die Metapher des „Objekts“ als Gegenstand offenbar ist. Versteht man den Begriff des Gegenstands im Sinne des Neuen Realismus als etwas, „worauf man sich mit wahrheitsfähigen Gedanken beziehen kann“ (vgl. oben), so passt das sehr treffend zur informatischen Modellbildung. Die im vorgestellten Lehrmaterial vorkommenden Diagramme entsprechen den Sinnfeldern des Neuen Realismus, in ihnen erscheinen die „Gegenstände“, d. h. die Objekte. Die informatische Modellbildung setzt ontologische Realitäten, um auf Ben-Ari (2001) zurückzukommen.

Die Freiheiten der Objektmodellierung
Man muss sich im Klaren darüber sein, dass man mit der „Stack“-Klasse Szenarien konstruieren kann, die sich nur schwer mit unserer Erfahrung vertragen. Schauen wir uns die folgende Stapelstruktur an:



Hier stapelt man dreimal ein und dasselbe „Book“-Objekt. Wohlgemerkt, es geht nicht um drei gleiche Bücher, sondern dreimal um ein und dasselbe Buch! Ich wüsste nicht, wie ich Ihnen das in der Realwelt anschaulich erklären sollte. Jetzt, wo wir in die Objektwelt eingetaucht sind, müssen wir akzeptieren, dass solch eine Modellbildung möglich geworden ist, vielleicht ist sie sogar für bestimmte Sachverhalte sinnhaft. Ein Abbild der Realität ist es nicht. Darin liegt die große Chance, aber auch der oft ungewollte Falschgebrauch informatischer Modellierung: Mit Kästchen und Pfeilen lässt sich allerhand zeichnen, allein die Klassendefinitionen erlegen den Objektarten bestimmte Pfeile und Pfeilziele auf. Die Freiheitsgrade sind so groß, dass man sich oft nicht einmal darüber bewusst ist, wie sehr man in Bezug auf eine Metapher oder in Anlehnung an Szenarien aus der Realwelt davon abweichen kann. Man muss Disziplin walten lassen im Gebrauch von Objektstrukturen oder sehr aufwendig versuchen, Strukturüberprüfungen einzubauen.

5 Abschließende Betrachtungen

Die erarbeitete Graph-Metapher hat bei meinen Studierenden einen Aha-Effekt ausgelöst – dem darf jedoch keine überhöhte Bedeutung zugeschrieben werden. Es wäre an der Zeit, eine Evaluation und eine systematische Studie vorzunehmen. Ein Ausgangspunkt könnte die Feldstudie von Ragonis und Ben-Ari (2005) sein, die sich speziell der Objektorientierung widmet. Die Autoren identifizieren 58 Schwierigkeiten, die Studierende mit der Objektorientierung haben, und geben ihre Lehrempfehlungen dazu ab. Man kann die Schwierigkeiten z. B. als Ausgangspunkt für einen zu entwickelnden Frage- oder Evaluationsbogen machen. Interessanterweise haben die Autoren in ihrem Kurs eine Entwicklungsumgebung namens BlueJ für den Java-Unterricht gewählt, die Klassen- und Objektbezüge visuell darstellt, was dennoch zu Fehlannahmen bei den Studierenden führte.

Alan Kay gilt als einer der Väter der Objektorientierung. Seine Arbeiten zur Programmiersprache *Smalltalk* aus den 1970er Jahren haben ihm unter anderem den „Nobelpreis der Informatik", den Turing-Award, eingebracht (Barnes, 2012). Kay bereut es im Nachgang, den Begriffs des „Objekts" geprägt zu haben (Kay, 1998). Auch sei die Idee der Klasse oder die Art der Syntax von *Smalltalk* nicht entscheidend: "The big idea is 'messaging' [...]" (ebd.). Kay, der ursprünglich Biologie studiert hat, hat Objekte als kommunizierende Wesen vor Augen gehabt, die Nachrichten empfangen und senden können. Diese Metapher ist sehr einfach; sie knüpft unmittelbar an die Erfahrung eines sozialen, verkörperten Wesens an. Mit ihr kann man die Konzepte der Objektorientierung synthetisieren in Form von Kommunikationsprotokollen, was *Smalltalk* sehr konsequent gemacht hat. Ein Programmierer ist in *Smalltalk* Teil des Systems und kann *Smalltalk* vollständig nach seiner Umsetzung „befragen", d. h. reflektieren. Der vorgestellten Graph-Metapher kann man durchaus eine Nachrichten-Metapher gegenüberstellen. Auch hierzu wären Untersuchungen reizvoll.

Dieser Beitrag hat am Beispiel der Programmierausbildung aufzuzeigen versucht, dass der Ansatz der Embodied Cognition wertvolle Impulse für die Entwicklung von Lehrmaterialien bietet, wobei die Chance und Möglichkeit besteht, didaktische Materialien und Vorgehen mit Hilfe der Lehr-Lernforschung experimentell zu evaluieren. Neu dürfte die wissenschaftstheoretische Verortung zum Neuen Realismus sein, der sich insbesondere im Zusammenhang mit der informatischen Modellbildung als passend zu erweisen scheint.

Literaturverzeichnis

Barnes, S. B. (2012). *Alan Kay. United States - 2003, Association for Computing Machinery*. Verfügbar unter https://amturing.acm.org/award_winners/kay_3972189.cfm [13.03.2019].

Ben-Ari, M. (2001). Constructivism in Computer Science Education. *Journal of Computers in Mathematics and Science Teaching 20* (1), 45–73.

Bergen, B. K. (2012). *Louder than words. The new science of how the mind makes meaning*. New York: Basic Books.

Crawford, M. B. (2016). *Die Wiedergewinnung des Wirklichen. Eine Philosophie des Ichs im Zeitalter der Zerstreuung*. Berlin: Ullstein eBooks.

Gabriel, M. (2013). *Die Erkenntnis der Welt. Eine Einführung in die Erkenntnistheorie* (4. Aufl.). Freiburg/München: Karl Alber.

Gabriel, M. (2018). *Warum es die Welt nicht gibt* (4. Aufl.). Berlin: Ullstein.

Gabriel, M. & Krüger, M. D. (2018). *Was ist Wirklichkeit? Neuer Realismus und Hermeneutische Theologie*. Tübingen: Mohr Siebeck.

Heublein, U., Ebert, J., Hutzsch, C., Isleib, S., König, R., Richter, J. & Woisch, A. (2017). *Motive und Ursachen des Studienabbruchs an baden-württembergischen Hochschulen und beruflicher Verbleib der Studienabbrecherinnen und Studienabbrecher* (DZHW Projektbericht 6|2017). Verfügbar unter https://www.dzhw.eu/pdf/21/BaWue_Bericht_gesamt.pdf [06.03.2019].

Heublein, U. & Schmelzer, R. (Oktober 2018). *Die Entwicklung der Studienabbruchquoten an den deutschen Hochschulen. Berechnungen auf Basis des Absolventenjahrgangs 2016* (DZHW-Projektbericht). Hannover: Deutsches Zentrum für Hochschul- und Wissenschaftsforschung (DZHW). Verfügbar unter https://www.dzhw.eu/pdf/21/studienabbruchquoten_absolventen_2016.pdf [06.03.2019].

Kay, A. C. (1998). *prototypes vs classes was: Re: Sun's HotSpot*. Verfügbar unter http://lists.squeakfoundation.org/pipermail/squeak-dev/1998-October/017019.html [13.03.2019].

Lakoff, G. & Johnson, M. (1999). *Metaphors we live by* [Nachdr.]. Chicago, Ill.: Univ. of Chicago Press.

Lakoff, G. & Núñez, R. E. (2000). *Where mathematics comes from. How the embodied mind brings mathematics into being*. New York, NY: Basic Books.

Qian, Y. & Lehman, J. (2017). Students' Misconceptions and Other Difficulties in Introductory Programming. *ACM Transactions on Computing Education 18* (1), 1–24. doi:10.1145/3077618

Quevedo-Torrero, J. U. (2009). Learning Theories in Computer Science Education. In *2009 Sixth International Conference on Information Technology: New Generations* (S. 1634–1635). IEEE.

Ragonis, N. & Ben-Ari, M. (2005). On understanding the statics and dynamics of object-oriented programs. In W. Dann, T. Naps, P. Tymann & D. Baldwin (Hrsg.), *Proceedings of the 36th SIGCSE technical symposium on Computer science education – SIGCSE '05* (S. 226–230). New York, USA: ACM Press.

Troelstra, A. S. (1999). From constructivism to computer science. *Theoretical Computer Science 211* (1–2), 233–252. doi:10.1016/S0304–3975(97)00172–2

Wehling, E. (2016). *Politisches Framing. Wie eine Nation sich ihr Denken einredet – und daraus Politik macht* (edition medienpraxis, Bd. 14, 1. Auflage). Köln: Herbert von Halem Verlag.

Autor

Dominikus Herzberg, Prof. Dr.-Ing.
Institut für Programmiersprachen und ihre Anwendung (IPA)
dominikus.herzberg@mni.th-mittelhessen.de